WWW.JAMK.FI

Digitaaliprojektorin seurantajärjestelmä

Veli-Matti Rajala

Opinnäytetyö
Toukokuu 2011

Tietotekniikan koulutusohjelma
Elektroniikka

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t)	Julkaisun laji	Päivämäärä
RAJALA, Veli-Matti Viljami	Opinnäytetyö	30.05.2011
	Sivumäärä 27	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
DIGITAALIPROJEKTORIN SEURANTAJÄRJESTELMÄ

Koulutusohjelma
Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
MIESKOLAINEN, Matti

Toimeksiantaja(t)
Finnkino Oy

Tiivistelmä

Työssä suunniteltiin Finnkinolle digitaaliprojektorin seurantajärjestelmä, joka sisältää sekä lukija- että vastaanotinyksikön. Työtä on tarkoitus soveltaa jopa kymmenelle digitaaliprojektorille samanaikaisesti. Laitteen on tarkoitus selventää ja helpottaa vahtimestarin ja koneenhoitajan digitaaliprojektorien seurantaa.

Lukijayksikkö sisältää kymmenen optoerottimilla erotettua sisääntuloa, joita luetaan Atmega8-mikrokontrollerilla. Mikrokontrolleri käsittelee tiedon ja lähettää sen vastaanotinyksikölle. Vastaanotinyksikkö käsittelee saamansa tiedon Atmega16-mikrokontrollerilla ja lähettää sen UARTlla ft-232-piirille, joka muuntaa tiedon USB-yhteensopivaksi. Tieto luetaan USB-väylästä tietokoneelta. Kummatkin yksiköt liitetään toisiinsa RJ-45-kaapelilla. Vastaanotinyksikön piirilevy on suunniteltu valmiiksi kymmenelle lukijayksikölle.

Työ vastasi suunnittelultaan sitä mitä siltä tullaan odottamaan. Kesken työn tapahtuneet laitemuutokset aiheuttivat toteutukseen ongelmia, jotka saatiin kuitenkin korjattua. Työn ajankäytössä oli myös haasteita.

Laitteella on paljon kehitysmahdollisuuksia. Ohjelmistoa on mahdollista kehittää tehokkaan mikroprosessorin ansiosta pidemmälle ja monipuolisemmaksi. SPI-väylä tuo mahdollisuuden lisätä kumpaankin laitteeseen erillisen lisäkortin.

Avainsanat (asiasanat)

mikrokontrolleri, digitaaliprojektori, automaatio

Muut tiedot

Author(s)
Rajala, Veli-Matti Viljami

Type of publication
Bachelor´s Thesis

Date
30.05.2011

Pages
27

Language
Finnish

Confidential

( ) Until

Permission for web publication
(X)

Title
MONITORING SYSTEM FOR DIGITAL PROJECTOR

Degree Programme
Information Technology

Tutor(s)
MIESKOLAINEN, Matti

Assigned by
Finnkino Oy

Abstract

The purpose of this thesis was to design a monitoring system for a digital projector for Finnkino Oy. The thesis includes the reader and the receiver unit and the work type is prototype. The device is intended to be applied to up to ten digital projectors at the same time. The purpose of this device is to clarify and make it easier for janitors and machine operators to monitor the digital projector.

The device has two units, the reader unit and the receiver unit. The reader unit comprises ten opto isolated inputs which are read with Atmega8 microcontroller. The microcontroller processes the received information and sends the data to the receiver unit. The receiver unit's Atmega16 microcontroller gathers all data from the reader unit and sends it with UART to ft-232 microchip. Ft-232 transfers the data to USB in a compatible format. The data is read from the USB bus by a computer. Both units are connected to each other by RJ-45 cable. The receiver unit´s printed circuit board is designed for ten reader units.

The design of the thesis was what was expected. Hardware changes in the middle of the thesis caused problems to the design but were corrected. There were also some issues of time management.

The device has a great deal of development potential. The program is fully adjustable because of the reprogrammable and powerful microcontroller. SPI bus brings a possibility to add another device card to both the reader and the transmission unit.

Keywords

Microcontroller, digital projector, automation

Miscellaneous

SISÄLTÖ

1 TYÖN LÄHTÖKOHDAT

Tämän opinnäytetyön tavoitteena oli suunnitella prototyyppi digitaaliprojektorin seurantalaitteesta. Laitteen tarkoitus on helpottaa ja yksinkertaistaa digitaaliprojektoreiden valvomista yhdestä paikasta ja auttaa teatterin vahtimestareita heidän työssään. Laite lukee digitaaliprojektorin lähettämät käskyt ja tulostaa ne tietokoneelle käytettäväksi tai toimii annettujen käskyjen perusteella.

Tavoitteena oli suunnitella kaksi laitetta: lukijayksikkö sekä vastaanotinyksikkö. Lukijayksikkö sisältää kaikki digitaaliprojektorinkäskyjen lukemiseen tarvittavat liittimet ja komponentit. Vastaanotinyksikkö taas sisältää kaiken elektroniikan lukijayksiköiden tiedon analysoimiseen ja sen tulostukseen tietokoneelle. Molemmat laitteet sisältävät mikrokontrollerin ja vastaanotinyksikkö sisältää myös FT232R-piirin, jonka tarkoitus on luoda USB -tiedonsiirtoyhteys laitteen ja tietokoneen välille. Molemmat laitteet kytketään toisiinsa RJ-45-kaapelilla.

Toteutusta varten oli muutamia vaatimuksia. Laite pitää voida yhdistää Dolby NA-10 Automaation kanssa ja laitteiden tulee toimia pitkienkin välimatkojen päästä. Virheitä ei saa tulla. Vastaanotinyksikön tulisi myös pystyä käsittelemään kymmentä lukijayksikköä ja tulostamaan näiden antama tieto tietokoneelle.

Marraskuun puolivälissä 2010 Finnkinon tekniikanryhmä kertoi koulutuksessa, että Dolby Laboratories on lopettanut Dolby NA-10 Automaatiojärjestelmän kehittämisen ja valmistamisen. Tämä johtaa siihen, että tulevaisuudessa kaikki digitaalisen projektorin sisäiset automaatiokäskyt pyritään välittämään verkkoa pitkin.

2 YMPÄRISTÖ

2.1 Atmel AVR-mikrokontrollerit

2.1.1 Yleistä

AVR:n perusarkkitehtuuri on kahden norjalaisen opiskelijan, Alf-Egil Bogen ja Vegard Wollan kehittelemä. Atmel osti AVR:n vuonna 1996, ja Bogen ja Vegard jatkoivat perusarkkitehtuurin kehittelemistä Atmelilla. AVR on yksi ensimmäisistä mikrokontrolleriperheistä, joka sisälsi uudelleen ohjelmoitavaa flash-muistia kertaalleen ohjelmoitavan muistin sijasta.

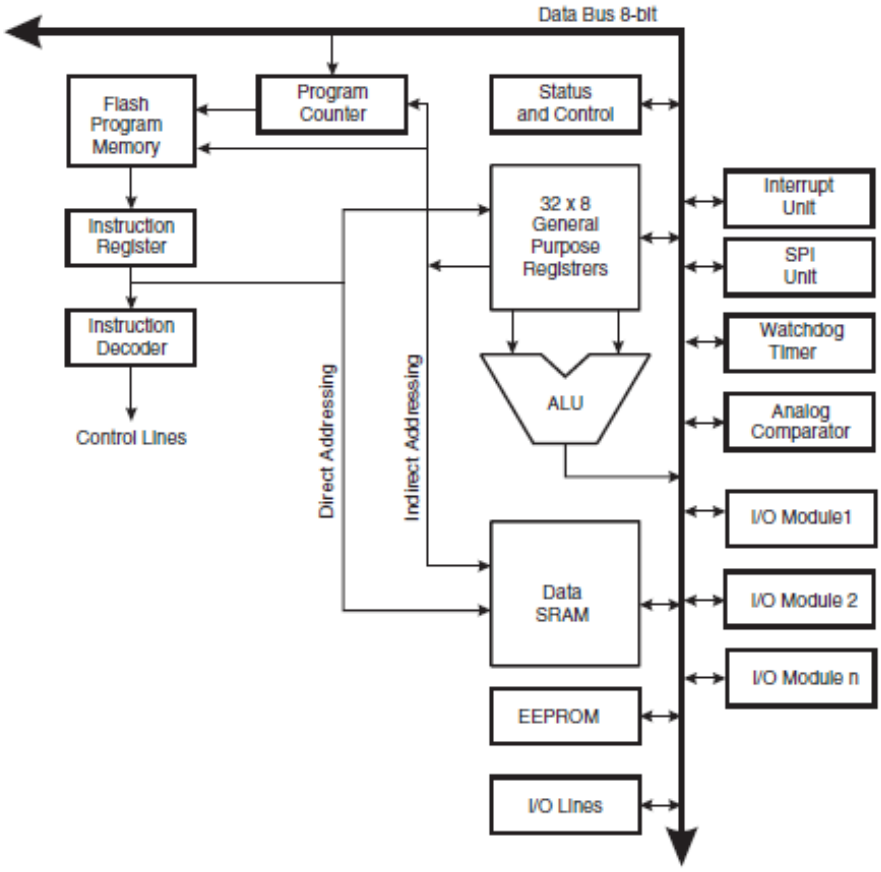
AVR-perheitä on viittä erilaista, joissa vaihtelee esim. kirjoitettavan muistin määrä, I/O-porttien lukumäärä ja halutut ominaisuudet. Perheitä ovat tinyAVR, megaAVR, XMEGA, Application Specific AVR ja FPSLIC™. Application Specific AVR eli sovelluskohtainen AVR-ryhmä sisältää valmiiksi tiettyyn tarkoitukseen suunniteltuja mikrokontrollereita, esim. LCD- tai USB-ohjaimia. FPSLIC™-mallit ovat taas FPGA-kotelolla olevia mikrokontrollereita, joissa on SRAM-muistia koodille, jota missään muussa mikrokontrollerissa ei ole. FPSLIC™-mallit voivat myös toimia jopa 50MHz taajuudella. (Atmel AVR. 2010.)

2.1.2 Arkkitehtuuri

AVR-arkkitehtuuri perustuu siihen, että jokaisessa mikropiirissä on oma muisti, joka poistaa ulkoisen muistin tarpeen useimmissa sovelluksissa. Melkein kaikissa piireissä on sarjaliikenneväylä jota voidaan hyödyntää yhdistäessä isompia EEPROM- tai Flash-muisteja. (Atmel AVR. 2010.)

Toimiakseen mahdollisimman tehokkaasti AVR käyttää Hardward-arkkitehtuuria mikrokontrollereissaan sekä erillisiä muisteja ja väyliä suorituskomennoille ja tiedolle.

Käskyt suoritetaan yksikerroksisella lukuhihnalla. Samaan aikaan kun käsky on suorituksessa, on seuraava jo haettu ohjelmamuistista. Tämä mahdollistaa käskyn suorittamisen jokaisella kellojaksolla. Ohjelmamuisti on uudelleen ohjelmoitavaa flash-muistia. Kuvio1 on Atmega 16 mikrokontrollerin lohkokaavio. (Atmega 16A tekniset tiedot. 2010.)



KUVIO 3 Atmega16 mikrokontrollerin arkkitehtuuri. (Atmega 16A tekniset tiedot. 2010.)

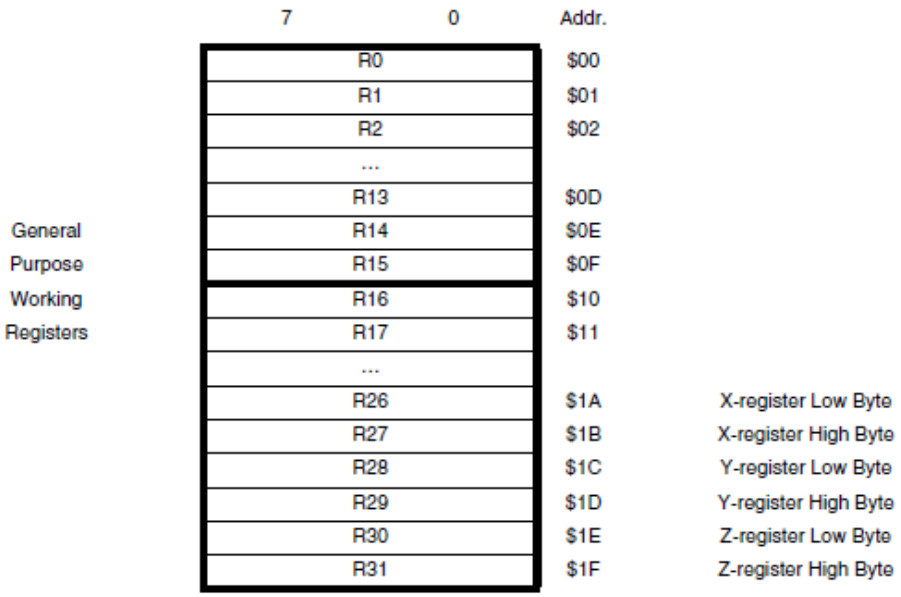
Tehokas AVR ALU (Arithmetic Logic Unit) on suorassa yhteydessä kaikkien 32 yleisrekisterin kanssa. Yhdessä kellojaksossa aritmeettiset operaatiot yleisrekisterien tai rekisterin tai vakion kanssa on suoritettu. ALU:n toiminta on jaettu kolmeen pääosaan: aritmeettinen, looginen ja bittikohtainen toiminta. (Atmega 16A tekniset tiedot. 2010.)

Tilarekisteri sisältää tiedon viimeksi suoritetusta aritmeettisesta käskystä. Tätä informaatiota voidaan käyttää prosessin muokkaamiseen, mikä mahdollistaa ehdollisten operaatioiden suorittamisen. Tilarekisteri päivittyy kaikkien ALU-operaatioiden jälkeen. Monessa tilanteessa tämä poistaa tarpeen käyttää vertailukäskyjä mistä seuraa nopeampaa ja vähemmän tilaa vievää koodia.

Yleisrekisteri on optimoitu AVR:n parannetulle RISC-tietokannalle. Tästä on saatu tarvittava nopeus ja joustavuus. Rekisteri tukee erilaisia tiedonsiirtotapoja:

- Yksi kahdeksanbittinen operandi ulos ja yksi kahdeksanbittinen operandi sisään
- Kaksi kahdeksanbittistä operandia ulos ja yksi kahdeksanbittinen operandi sisään
- Kaksi kahdeksanbittistä operandia ulos ja yksi kuusitoistabittinen operandi sisään
- Yksi kuusitoistabittinen operandi ulos ja yksi kuusitoistabittinen operandi sisään.

Rekistereillä R26-R31 on joitain lisättyjä toimintoja niiden normaalin toimintaa liittyen. Nämä rekisterit ovat kuusitoistabittisiä osoittajia data-alueeseen. Näitä kutsutaan X-, Y-, ja Z-rekistereiksi (ks. kuvio 2). Näillä rekistereillä on toimintoja eri ositustiloissa, kuten automaattinen lisäys ja vähennys sekä kiinteä siirtymä. (Atmega 16A tekniset tiedot. 2010.)



KUVIO 4: AVR mikrokontrollerin yleisrekisteri. (Atmega 16A tekniset tiedot. 2010.)

2.1.3 Muistit

AVR-arkkitehtuurissa on yleisesti kolmea eri muistia. Flash-ohjelmamuistia, SRAM-datamuistia ja pitkäaikaista EEPROM-muistia, jota piiri käyttää tiedon tallentamiseen. Atmega-perheen piireissä voi olla flash-muistia 4 – 256 kB. Atmega16 sisältää 16 kilobittiä uudelleen ohjelmoitavaa flash-muistia ohjelman säilytykseen. Kaikki AVR:n käskyt ovat 16 tai 32 bittiä leveitä, flash-muisti on jaettu 8 k x 16-bittiseen osaan. Ohjelman turvallisuutta ajatellen flash-muisti on jaettu kahteen osaan: esilataajan ja sovellusohjelmanosioihin. Flash-muisti kestää vähintään 10 000 kirjoitus / tyhjennyssykliä. (Atmega 16A tekniset tiedot. 2010.)

SRAM-muisti sisältää rekisteritiedoston, I/O-muistin ja datamuistin. Ensimmäiset 32 osoitetta sisältävät yleiset rekisteritiedostot, seuraavat 64 osoitetta ovat I/O-rekisterit ja loppu on datamuistia. (Atmega 16A tekniset tiedot. 2010.)

Atmega16:ssa on EEPROM-datamuistia 512 bittiä. Muisti on organisoitu erilliseksi data-alueeksi, jonka jokaista yksittäistä bittiä voi lukea tai kirjoittaa. EEPROM-muistilla on 100 000 kirjoitus/tyhjennyssyklin kestävyys. (Atmega 16A tekniset tiedot. 2010.)

2.1.4 AVR-mikrokontrollerin ominaisuuksia

AVR mikrokontrolleriin on integroitu monia ominaisuuksia perheestä riippuen. Jokaisessa mikrokontrollerissa on sisäänrakennettuna esim. sisäinen oskillaattori, käyttömuistia ja eri väylämahdollisuuksia tiedonsiirtoon.

Atmega16 on monipuolinen mikrokontrolleri. Siinä on 16 kilotavua ohjelmoitavaa flash-muistia, 1024 tavua SRAM-käyttömuistia ja 512 tavua EEPROM-muistia tietojen tallentamiseen. Atmega16 sisältää myös 8-kanavaisen 10-bittisen analogi/digitaali-muuntimen, neljä pulssileveysmoduloitua kanavaa, analogisen komparaattorin, JTAG-rajapinnan piirin testaukseen ja vianetsintään. Tiedonsiirtoa varten löytyy SPI, USART ja TWI. Mikrokontrolleria voidaan käyttää 2.7 -5.5 V käyttöjännitteellä. 16

megahertsin taajuudella saadaan jopa 16 MIPS:in suorituskyky. (Atmel Parametric Product Table. 2010)

2.2 UART

UART on universaali asynkroninen lähetys/vastaanotinyksikkö, joka muuntaa tietoa rinnakkaisen ja sarjamuotoisen tiedon välillä. UARTia käytetään usein muiden kommunikaatiostandardien kanssa, kuten esimerkiksi RS-232 kanssa. Nykyisissä mikropiireissä on USART, joka pystyy keskustelemaan myös synkronisesti.

UART lähettää datan sarjamuotoisena yhtä johdinta pitkin bitti kerrallaan, ja vastaanottaja kokoaa kasaan toisessa päässä. Tiedon siirtäminen yhtä johdinta pitkin on paljon tehokkaampaa kuin rinnakkaismuotoinen siirtäminen montaa johdinta pitkin. Jokaisessa UARTissa on siirtorekisteri, joka on keskeisessä osassa muunnettaessa sarjamuotoista tietoa rinnakkaiseen muotoon ja toisinpäin. USART ei yleensä generoi itse tarvittavia signaalitasoja, vaan sen hoitaa erillinen liitäntärajapinta. Ulkoiset signaalit voivat olla monessa eri muodossa. Esimerkkejä standardijännitesignaaleista ovat RS-232, RS-422 ja RS-485. (Universal asynchronous receiver/transmitter. 2011.)

Tiedonsiirto voi olla ”full duplex” (vastaanottaa ja lähettää samanaikaisesti) tai ”half dublex” (laitteet ottavat vuoroja lähetykseen ja vastaanottamiseen). Vuodesta 2008 UART on ollut yleisesti käytössä laitteiden välisessä kommunikaatiossa RS-232 kanssa. Asynkronisessa tiedonsiirrossa UART lähettää ”start”-bitit, 5-8 databittiä, vähiten merkitsevä ensiksi, vaihtoehtoisen pariteettibitin, ja sitten yksi tai puolitoista tai kaksi ”stop”-bittiä. Aloitusbitti on vastakkainen kuin datalinjan joutotila. Lopetusbitti on samanlainen kuin datalinjan joutobitti, jolloin se mahdollistaa tauon seuraavan lähetyksen alkuun. (Universal asynchronous receiver/transmitter. 2011.)

Synkronisessa tiedonsiirrossa kellosignaali saadaan erillisestä lähteestä eikä datavirrasta. Tällöin ei erillistä aloitus- eikä lopetusbittiä tarvita. Tämä nostaa tiedonsiirron tehokkuutta sitä hyödyntävissä kanavissa, koska liikkuva informaatio on dataa eikä ylimääräisiä reunusbittejä. Asynkroninen lähetys ei lähetä mitään joutotilassa, mutta

synkronisen on lähetettävä jatkuvasti ”pad”-merkkiä pitääkseen laitteiden välistä synkronointia päällä. Yleensä tämä on ASCII-merkistön ”SYN”-merkki. Yleensä laitteet pystyvät tekemään tämän automaattisesti. (Universal asynchronous receiver/transmitter. 2011.)

2.3 Dolby NA-10 Network Automation

NA-10 on Dolby Laboratoriesin kehittämä automaatiojärjestelmä, jolla on yksinkertaista ohjata jo olemassa olevia elokuvateatteritoimintoja, kuten valoja, verhoja, maskeja ja eri ääniformaatteja. Laite antaa käyttäjälle mahdollisuuden käyttää ”cue” – käskyjä suoraan elokuvan soittolistalta. Tämä mahdollistaa filmiprojektorin automaation käytön suoraan digitaalisen projektorin hallintaympäristöstä ja automatisoi täysin elokuvan tarvittavat toiminnot. NA-10 voidaan myös liittää rinnakkain jo olemassa olevan automatiikan kanssa. (Dolby Laboratories Inc. 2010.)

Automaation hallitsemiseksi NA-10-automaatiojärjestelmä pitää sisällään käyttöympäristön, jossa käyttäjä pystyy tekemään ”cue” eli ”teippejä”. Teipit ovat ohjelmallisia viittauksia tiettyihin releisiin laitteen sisällä. Käyttöympäristössä määritellään releen tila, vetämisen pituus, ohjelmallisesti mihin ryhmään teippi kuuluu ja onko laitteen käynnistyksessä tarpeen pitää relettä päällä, eli ohjelmallinen teippi on tietyn releen toiminta-asetukset.

2.4 FT-232b USB-UART mikropiiri

Ft-232b on FTDI Chipin valmistama USB – UART-muunnospiiri. Yleisesti ft-232b-piiriä käytetään muun muassa USB – RS232-muunnokseen, nykyaikaistettaessa vanhempia tiedonsiirtotapoja, matkapuhelinten USB-tiedonsiirrossa ja USB-kortinlukijoissa. Piirissä on sisäänrakennettuina toimintoina muun muassa

- 4.35 V – 5.25 V käyttöjännite (USB-jännite)
- sisäänrakennettu 6 MHz – 48 MHz kellojakaja
- täydelliset kättely- ja rajapintasignaalit modeemille

- tukee 7/8-bittistä tiedonsiirtoa, ½ lopetusbittiä ja parillista, paritonta, merkki, tyhjää tai kokonaan ilman olevaa pariteettia
- tiedonsiirtonopeuksia:
    - 300 -> 3M baudia (TTL)
    - 300 -> 1M baudia (RS232)
    - 300 -> 3M baudia (RS422/RS485)
- 384-tavuinen vastaanottopuskuri / 128-tavuinen lähetyspuskuri
- USB 1.1 ja USB2.0 yhteensopiva

Työssä on käytetty ft-232b-piiriä tuottamaan UART:sta USB-käyttöliittymä tietokoneelle. (FTDI Chip. 2010.)

3 TOTEUTUS

3.1 Yleistä

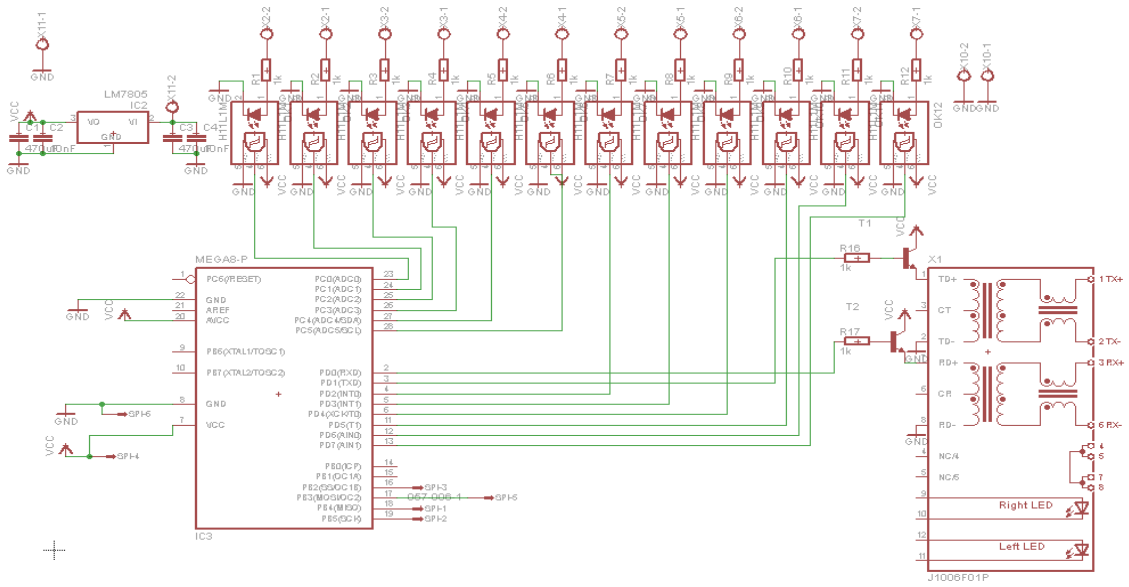
Työhön kuului kaksi piirilevyä: lukija ja vastaanotin. Vastaanotin toimii työn aivoina, joka tekee suurimman työn tiedon käsittelyssä, ja lukija vain seuraa NA-10-releiden tilaa ja lähettää vastaanottimelle tiedon muutoksesta. Vastaanottimen tulisi pystyä käsittelemään kymmentä lukijaa samanaikaisesti. Prototyypin oli tarkoitus käsitellä vain yhtä.

Laitteet ovat kytkettyinä RJ-45-kaapelilla toisiinsa. Kaapelissa on tarkoitus siirtää tietty määrä pulsseja tiettyä tietoa vastaan. Pulssit ovat pitkiä ja pulssien välit ovat pitkiä. Tällä pyritään poistamaan turhia liipaisuja pitkillä kaapeleilla. Ohjelmallisesti toteutettiin virheenkorjaus, jolla eliminoidaan turhat liipaisut pulssissa tai niiden välissä.

3.2 Kytkentäkaaviot

Lukija toimii työn informaation tuottajana vastaanotinyksikölle. Lukija sisältää Atmega 8-mikrokontrollerin, 12 optoerotinta, 5 V:n jänniteregulaattorin ja RJ-45-liittimen. Kytkennässä oleva jänniteregulaattori tuottaa piirin tarvitsevan 5V:n käyttöjännitteen laitteeseen sisään tulevasta jännitteestä, joka on 12 V. SPI:lle varatut MOSO, MISO, SS ja SCK ovat käytössä piirin ohjelmoimiseen ja päivittämiseen piirin ISP-väylää pitkin.

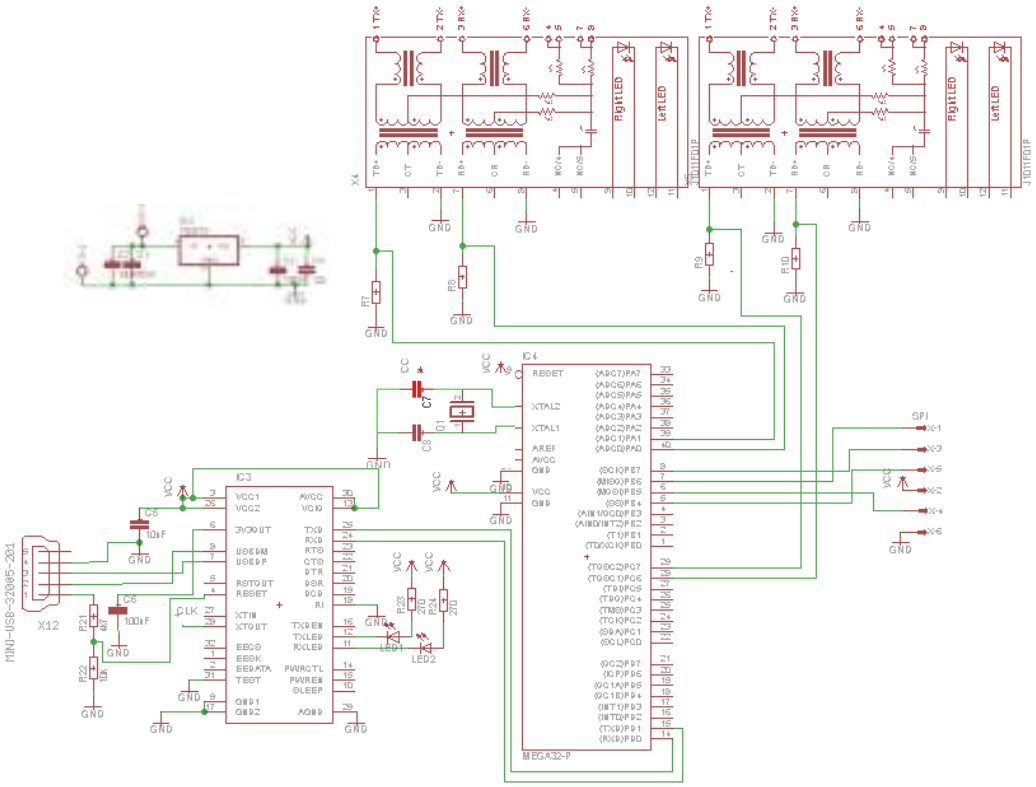
Laitteessa jokainen NA-10 tuleva sisääntulo on erotettu optoerottimilla itse kytkennästä, jotta se estäisi ylimääräisten häiriöiden syntymisen järjestelmään. Piirin ulostulona toimii kaksi linjaa, jotka menevät vastuksen kautta transistorille. Transistorit toimivat kytkiminä, jotka antavat siirtolinjaan suuremman virran. Niillä varmistetaan tarvittava energiamäärä tiedonsiirtoon pitkilläkin matkoilla. Lukijan kytkentäkaavio näkyy kuviossa 3.



KUVIO 3. Lukijan kytkentäkaavio

Vastaanotinyksikkö sisältää Atmega16-mikropiirin, 12 MHz:n kiteen, ft-232-sarjamuunnospiirin, kymmenen RJ-45-liitintä lukijayksiköille, 5V:n jänniteregulaattorin sekä mikro-USB-liittimen. Vastaanotinyksikön on tarkoitus kerätä tiedot lukijayksiköltä ja prosessoida tiedot annettujen ohjeiden mukaisesti ja lähettää haluttu tieto UARTilla ft-232-piirille, joka muuntaa tiedon USB-yhteensopivaksi. ISP-väylää käyte-

tään piirin ohjelmoimiseen ja päivittämiseen kun siihen tulee tarvetta, kuten lukijassakin. Vastaanotinyksikön kytkentäkaavio on esitetty kuviossa 4.

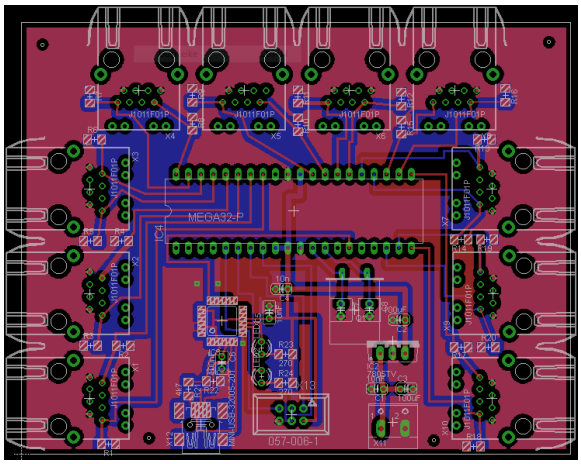


KUVIO 4 Vastaanotinyksikkö vain kahdella RJ-45 liittimellä kuvattuna

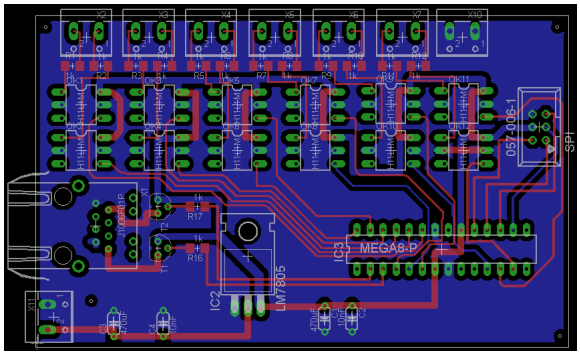
3.3 Työn kokoaminen

Piirilevyjen suunnittelussa tuli ottaa huomioon muutama seikka. Koska laite ei ole fyysisesti suuri, laite tuli suunnitella liittimien suhteen kummassakin laitteessa mahdollisimman järkevästi. Liittimet sijoitettiin laitteen reunoille, mutta johtojen suunta pyrittiin pitämään mahdollisimman samana toisiinsa nähden. Mikropiirien ja jännitteregulaattorien suotokondensaattorit tuli sijoittaa mahdollisimman lähelle itse piiriä, millä pyrittiin estämään rasituksessa ilmenevät jännitenotkahdukset.

Piirilevyjen ylimääräinen tila täytettiin maatasolla häiriöiden välttämiseksi. Kuvioissa 5 ja 6 on erotettu punaisella värillä pintapuolen johtimet ja kuparitäyttö. Pohjapuolen johtimet ja kuparitäyttö näkyy sinisenä. Lukijassa ei ole yläpuolella kuparitäyttöä lainkaan. Komponentit on sijoitettu kumpaankin vain yläpuolelle piirilevyä.



KUVIO 5 Vastaanotinyksikkö komponenttipuolelta kuvattuna



KUVIO 6 Lukijayksikkö komponenttipuolelta kuvattuna

Lukijayksikkö tulee NA-10 automaatiojärjestelmän sisälle tai vaihtoehtoisesti koko järjestelmän metallirungon sisään. Lukijayksikköön ei ole tarkoitus tehdä koteloa. Vastaanotinyksikkö tulee pidettäväksi pöydällä, joten se on tarkoitus suoja kotelolla, johon leikataan aukot liittimille ja merkkivaloille.

3.4 Mikrokontrollerin ohjelmointi

Mikrokontrollerit ohjelmoidaan AVR MKII laitteella. Laitteella pystyy ohjelmoimaan mikrokontrollerin flash- ja EEPROM-muistin sekä muokkaamaan sulakebittejä mikrokontrollerin omaa ISP (In-System Programmer) väylää pitkin. Ohjelmointi tapahtuu erillisellä adapterilla. Kummassakin piirilevyssä adapteri on suunniteltu suoraan piirilevylle tulevia ohjelmistopäivityksiä varten. Laite ei tarvitse ulkoisia komponentteja, ainoastaan piirikohtaisen adapterin, jolla tarvittavat pinnit yhdistetään ohjelmointilaitteeseen. (AVRISP mkII User Guide. 2010.)

4 OHJELMAN TOIMINTA

Työssä on kaksi laitetta. Lukija ottaa vastaan ulkoisia pulsseja ja välittää informaation vastaanotinyksikölle. Vastaanotinyksikkö käsittelee jokaiselta lukijalta saadun tiedon, tekee tarvittavat ohjelmalliset toimenpiteet ja lähettää halutun tiedon ft-232 piirin avulla USB:n kautta tietokoneelle. Tietokoneella tulostetaan tieto käyttäjälle.

4.1 Lukijan ja vastaanotinyksikön välinen tiedonsiirto

Lukijassa on kaksitoista input–porttia automaatiojärjestelmälle, joista jokaisella on oma tunnuksensa. Lukijassa on tarkoitus ainoastaan lukea pulsseja, eikä tuoda mitään signaaleja itse automaatiojärjestelmään ja näin ollen pyritään estämään kaikki häiriöt laitteesta ulospäin. Lukijan ja vastaanottoyksikön välinen tiedonsiirto tapahtuu RJ-45 liittimen kautta kumpaankin suuntaan. Vastaanotinyksikössä on kymmenen RJ-45 liitintä, joissa jokaisessa on kaksi porttia varattuna jokaista lukijaa kohti tiedonsiirtoon.

Ohjelmallisesti tietyn portin tunnistus on toteutettu (lukija & vastaanotin v0.5) pulssien määrällä. Esimerkiksi jos porttiin neljä tulee pulssi, niin lukija lähettää neljä tietyn pituista pulssia vastaanotinyksikölle. Vastaanotinyksikkö pystyy identifioimaan informaation laskemalla pulssit tietystä RJ-45 liittimestä, joka on nimetty tietyksi lukijaksi.

Vastaanotinyksikössä on virheenkorjauksena laskurit joissa signaali pitää pysyä ylhäällä ja alhaalla tietty toleranssiaika pulssin pituudesta jolloin se lasketaan ykköseksi tai nollaksi ennen kuin lasketaan seuraava pulssi. Kun kaikki pulssit ovat siirtyneet, tulee pitkä viive, jolloin vastaanotin yksikkö ymmärtää että lähetys on loppunut. Tällöin vastaanotinyksikkö lähettää toista linjaa pitkin vastaanotettua lähetystä vastaavan pituisen pulssin lukijalle. Lukija tarkastaa pulssin pituuden tiettyjä toleranssiarvo-

ja noudattaen. Jos lähetys on mennyt perille, lukija kuittaa pulssilla samaan linjaan vastaanotinyksikölle, joka tällöin jatkaa informaation käsittelyä normaalisti. Jos pulssit eivät ole siirtynyt ehjänä perille, lukija ei kuittaa lähetystä, vaan aloittaa saman pulssisarjan lähetyksen vastaanotinyksikölle uudestaan. Tämä toistuu niin kauan kuin kummassakin on varmistus informaation oikeellisuudesta. Liitteenä on lukijan ja vastaanottoyksikön ohjelmakoodit. (Ks. liite 1 ja liite 2.) Ohjelmakoodeissa on käytetty apuna CodeVisionAVR:n automaattista ohjelmakoodin generointi työkalua.

Tiedonsiirto laitteen ja lukijan välille on pystytty toteuttamaan yksinkertaisesti, koska ei ole tarvetta siirtää suurta määrä informaatiota laitteiden välillä. Kuitenkin työssä on käytetty mikrokontrolleria mahdollistaakseen lukijan päivittämisen ja toimintojen muuttamisen, joka olisi lähes mahdotonta jos samat toiminnot olisi tehty itse komponenteilla. Ohjelma lukijaan sekä vastaanotinyksikköön on tehty CodeVision AVR ohjelmalla. Mikrokontrollereiden sulakebittejä muokataan AVR Studio 4 ohjelmaa käyttäen.

4.2 Vastaanottimen ja tietokoneen välinen tiedonsiirto

Tiedonsiirto vastaanotinyksikön ja tietokoneen välillä on toteutettu FT-232 piiriä hyväksikäyttäen. Mikrokontrollerilla lähetetään UART:illa FT-232 piirille informaatiota, joka muuttaa sen USB yhteensopivaksi jolloin laite on helppo liittää tietokoneeseen. FT-232 piirin ajuri muuntaa USB väylän tietokoneen sarjaliikenneportiksi, jonka jälkeen kyseisestä portista hyperterminaali sovellus lukee (Vastaanotinyksikkö v0.1) lähetetyn tiedon ja tulostaa näytölle.

5 TESTAUKSEN SUUNNITTELU

Laite testataan tekniikaltaan ja ohjelmistoltaan erilaisilla testausmenetelmillä. Piirilevyjen kuparivedot tarkastetaan manuaalisesti yleismittarilla. Käsintehdyssä piirilevyssä voi ilmetä kuparivetojen liiallista tai liian vähäistä syöpymistä. Tästä voi seurata huonot tai ylimääräiset kontaktit piirilevyllä.

Mikropiirien toimivuutta testataan juottamisen jälkeen erilaisilla yksinkertaisilla loogisilla operaatioilla. Testaukset diagnosoidaan RJ-45 liittimeen liitettävällä piirilevyllä, joka koostuu muutamasta vastuksesta ja hohtodiodista. Näin saadaan varmuus piirin toimivuudesta.

Suurimpia oletettavissa olevia ongelmia laitteessa on lukijan ja vastaanottoyksikön välinen tiedonsiirto. Näiden kahden väliset etäisyydet voivat olla monia kymmeniä metrejä. Testauksessa on tarkoitus demonstroida samoja olosuhteita kuin laitteen toimintaympäristössä. Kaapeleiden pituudet ja niiden väliset liittimet, voivat tuoda häiriöitä tiedonsiirtoon.

6 YHTEENVETO

Työ vastasi suunnittelultaan hyvin sitä mitä siltä tullaan tarvitsemaan. Kesken työn tapahtuvat laitemuutokset aiheuttivat toteutukseen ongelmia, jotka saatiin kuitenkin korjattua piirilevyn piirustuksiin. Työn ajankäytössä oli myös suuria haasteita. Työ toi ymmärrystä, miten paljon jonkun asian korjaaminen voi viedä aikaa riippuen siitä, missä vaiheessa tuote on elinkaartaan.

Laitteella on paljon kehitysmahdollisuuksia. Työhön voi liittää langattoman lähettimen ja vastaanottimen, minkä jälkeen digitaaliprojektorien seuranta muuttuu vielä helpommaksi. Ohjelmistoa on mahdollista kehittää tehokkaan mikroprosessorin ansiosta pidemmälle ja monipuolisemmaksi. On mahdollista ottaa lukijasta riippumattomat toiminnot mukaan ja laajentaa niitä tarpeen mukaan. Ohjelmointiväylä ISP:n adapteria voidaan suoraan käyttää SPI-väylän käyttöön. SPI-väylä tuo mahdollisuuden lisätä kokonaan uusi lisäkortti kumpaankin laitteeseen.

LÄHTEET

Atmel AVR. 2010 Wikipedia: Vapaa tietosanakirja. Viitattu 15.9.2010. Muokattu 10.9.2010. http://en.wikipedia.org/wiki/Atmel_AVR

Atmega 16A tekniset tiedot. 2010. Viitattu 2.10.2010. http://www.atmel.com/dyn/resources/prod_documents/doc8154.pdf

AVR Solutions. 2010. Teknillisiä ominaisuuksia sisältävä sivu. Viitattu 16.09.2010. http://www.atmel.com/products/AVR/megaavr.asp?family_id=607

Atmel Parametric Product Table. 2010. Listattu teknilliset ominaisuudet. Viitattu 20.9.2010. http://www.atmel.com/dyn/products/param_table.asp?family_id=607&OrderBy=part_no&Direction=ASC

Universal asynchronous receiver/transmitter. 2011. Wikipedia: Vapaa tietosanakirja. Viitattu 16.5.2011. Muokattu 13.3.2011. http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter

Dolby Laboratories Inc. 2010. Dolby Laboratories ilmoitus. Viitattu 29.11.2010. http://investor.dolby.com/releasedetail.cfm?ReleaseID=155919

FTDI Chip. 2010. FT-232BL/BQ Piirin tekniset tiedot. Viitattu 25.11.2010. http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232BL_BQ.pdf

AVRISP mkII User Guide. 2010. AVRISP mkII laitteen käyttöohje. Viitattu 10.12.2010. http://www.atmel.com/dyn/resources/prod_documents/AVRISPmkII_UG.pdf

LIITTEET

Liite 1 Lukijayksikön ohjelmistokoodi

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Lukija
Version : 0.5
Date    : 23.9.2010
Author  : Veli-Matti
Company : Rajala
Comments:
Opinnäytetyön lukija osan ohjelmistokoodi.


Chip type           : ATmega8
Program type        : Application
Clock frequency     : 4,000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega8.h>
#include <stdio.h>
#define F_CPU 4000000UL
#include <delay.h>
#define _BV(bit)(1 << (bit))

char luku;
unsigned int ala=0;
unsigned int yla=0;
unsigned int lippu=1;
unsigned int tila=0;
int x=0,y=0,z=0,o=0;
unsigned char rele;


void pulssikone (int luku);
int muisti (int luku);
interrupt [TIM1_CAPT] void timer1_capt_isr(void);
interrupt [EXT_INT1] void ext_int1_isr(void);

void main(void)
{

 #asm("sei")
```

```
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=Out Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=0 State0=T
PORTD=0x00;
DDRD=0x04;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
```

```
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;


// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;


while (1)
    {

    if (PINC.0)
      {
      luku = 3;
      muisti(luku);
      }
    if (PINC.1)
      {
      luku = 4;
      muisti(luku);
      }
    if (PINC.2)
      {
      luku = 5;
      muisti(luku);
      }
    if (PINC.3)
      {
      luku = 6;
      muisti(luku);
      }
    if (PINC.4)
      {
      luku = 7;
      muisti(luku);
      }
    if (PINC.5)
      {
      luku = 8;
      muisti(luku);
      }
    if (PIND.0)
      {
      luku = 9;
      muisti(luku);
      }
    if (PIND.1)
      {
      luku = 10;
```

```
      muisti(luku);
      }
    if (PIND.3)
      {
      luku = 11;
      muisti (luku);
      }
    if (PIND.4)
      {
      luku = 12;
      muisti(luku);
      }
    if (PIND.5)
      {
      luku = 13;
      muisti(luku);
      }
    if (PIND.6)
      {
      luku = 14;
      muisti(luku);
      }
    if (PIND.7)
      {
      luku = 15;
      muisti(luku);
      }

      while (x==!0)
                              {

                                      if (lippu==1)          //lupa ok
                              {
                              tila=yla - ala;

                              if (tila==x)
                              {
                              PORTD |= _BV(1);
                              delay_ms(x);                    //x:n verran pinni ylhäällä
                              PORTB &= ~_BV(1);
                              lippu=0;
                              x=0;                                            //x:n nollaus
                              }


                    }
              }
      }
}
void pulssikone(int luku)
{
      while (luku > 0)
            PORTB |= _BV(3);
            delay_ms(1)
            PORTB &= ~_BV(3);
            delay_ms(1)
```

```
            luku--;
      return;
}
x muisti (int luku)
{
    if (x==0)

       x=y;

    else
    if (x~0, y==0)

       y=z;
    else
    if (x~0, y~0, z==0)

       z=o;
    else
    if (x~0, y~0, z~0, o==0)

       o==luku;
    else (x=0)
    return x;
}

interrupt [TIM1_CAPT] void timer1_capt_isr(void)
}
if (MCUCR==0x03)
      {
      TCNT0=0;                  //nollaa laskurin
      ala=TCNT0;
      lippu=0;
      MCUCR=0x02;               //asetetaan laskevalle reunalle
      }
else
      {
      yla=TCNT0;                //laskevalle reunalle kertynyt aika
      lippu=1;                  //lupalippu saatu == mittaus tulokset saatu
      MCUCR=0x03;               //asetetaan nousevalle reunalle
      }
}

interrupt [EXT_INT1] void ext_int1_isr(void)
{
      if (MCUCR==0x03)
            {
            TCNT0=0;             //nollaa laskurin
            ala=TCNT0;
            lippu=0;
            MCUCR=0x02;       //asetetaan laskevalle reunalle
            }

      else
            {
            yla=TCNT0;          //laskevalle reunalle kertynyt aika
            lippu=1;            //lupalippu saatu == mittaus tulokset saatu
```

```
            MCUCR=0x03;     //asetetaan nousevalle reunalle
            }
                            //MCUCR=0x03;   //nousevalla reunalla
                             //MCUCR=0x02;  //laskevalla reunalla
}
```

Liite 2 Keskeneräinen vastaanottoyksikön ohjelmisto-koodi yhdellä lukijalla

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Vastaanotinyksikkö
Version : 0.5
Date    : 25.9.2010
Author  : Veli-Matti Rajala
Company : -
Comments:
Opinnäytetyön vastaanottoyksikön koodi

Chip type           : ATmega16
Program type        : Application
Clock frequency     : 12,000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>
#define F_CPU 12000000UL
```

```
#include <delay.h>
#include <stdio.h>

#define LOW 0
#define HIGH 1

#define INPUT(port,pin) DDR ## port &= ~(1<<pin)
#define OUTPUT(port,pin) DDR ## port |= (1<<pin)
#define CLEAR(port,pin) PORT ## port &= ~(1<<pin)
#define SET(port,pin) PORT ## port |= (1<<pin)
#define TOGGLE(port,pin) PORT ## port ^= (1<<pin)
#define READ(port,pin) (PIN ## port & (1<<pin))

// Declare your global variables here
void InitUART( unsigned char baudrate );
unsigned char ReceiveByte( void );
void TransmitByte( unsigned char data );
int laheta_varmistus (int);

void main(void)
{
unsigned char paketti= 0;
int laskuri1,laskuri2 = 0;
int laskulupa, i = 0;

PORTA=0x00;
DDRA=0xAA;

PORTB=0x00;
DDRB=0x00;

PORTC=0x00;
DDRC=0x55;
```

```
PORTD=0x00;
DDRD=0x50;

TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

MCUCR=0x00;
MCUCSR=0x00;

TIMSK=0x00;

UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
```

```
UBRRL=0x4D;

ACSR=0x80;
SFIOR=0x00;

InitUART( 6 );
while (1)
{
    if (READ(A,7)==HIGH)
    {
     i++;
     laskulupa = 1;
     paketti = kone_paalla/n
    };
    else (READ(A,7) == LOW)
    {
     laskulupa = 0;
     laskuri1 = 0;
    };
     }
    if (laskulupa == 1)
    {
     laskuri1++;
     laskuri2++;
    };

    if (laskuri1 >= 300000)
    {
     laheta_varmistus(i);
     i=0;
       if((laskuri2 >= 270000) && (laskuri2 <= 320000))
    {
     TransmitByte(paketti);
```

```
    };
};
}
void laheta_varmistus (int i)
{
      PORTA |= _BV(6);
      delay_ms(i);
      PORTA &= ~_BV(6);
 }
void InitUART( unsigned char baudrate )
{
                  UBRRL = baudrate;
                  UCSRB = (UCSRB | _BV(RXEN) | _BV(TXEN) );      }
unsigned char ReceiveByte( void )
{
      while ( !(UCSRA & (_BV(RXC))) );
      return UDR;
}
void TransmitByte( unsigned char data )
{
      while ( !(UCSRA & (_BV(UDRE))) );
      UDR = data;
}
```